

# CreateProcess-03

Use restrictive permissions when creating a new process

Sean Barnum, Digital, Inc. [vita<sup>1</sup>]

Copyright © 2007 Digital, Inc.

2007-03-20

## Part "Original Digital Coding Rule in XML"

Mime-type: text/xml, size: 10616 bytes

<b>Attack Category</b>	• Privilege Exploitation		
<b>Vulnerability Category</b>	• Process management • Privilege escalation problem		
<b>Software Context</b>	• Threads and Processes		
<b>Location</b>			
<b>Description</b>	<p>Do not give write permission when creating a new process.</p> <p>CreateProcess() and related functions allow one to spawn a new process. When a process is created, one can specify access rules for that process. Processes with loose access rules pose a security risk.</p>		
<b>APIs</b>	<b>FunctionName</b>	<b>Comments</b>	
	CreateProcess		
	CreateProcessA	ANSI implementation	
	CreateProcessW	Unicode implementation	
	CreateProcessAsUser		
	CreateProcessWithLogonW		
<b>Method of Attack</b>	If an attacker can get a handle to your process with some sort of write access, he can call functions such as WriteProcessMemory() to rewrite parts of the process to effect some sort of stack smashing attack.		
<b>Exception Criteria</b>			
<b>Solutions</b>	<b>Solution Applicability</b>	<b>Solution Description</b>	<b>Solution Efficacy</b>
	When creating a new process.	Do not set permissions of child process so as to allow an attacker to rewrite your	Effective if done properly. Requires understanding of security model.

1. <http://buildsecurityin.us-cert.gov/bsi-rules/35-BSI.html> (Barnum, Sean)

	child's memory space.
<b>Signature Details</b>	<pre> BOOL CreateProcess( LPCTSTR lpApplicationName, LPTSTR lpCommandLine, LPSECURITY_ATTRIBUTES lpProcessAttributes, LPSECURITY_ATTRIBUTES lpThreadAttributes, BOOL bInheritHandles, DWORD dwCreationFlags, LPVOID lpEnvironment, LPCTSTR lpCurrentDirectory, LPSTARTUPINFO lpStartupInfo, LPPROCESS_INFORMATION lpProcessInformation );  BOOL CreateProcessAsUser( HANDLE hToken, LPCTSTR lpApplicationName, LPTSTR lpCommandLine, LPSECURITY_ATTRIBUTES lpProcessAttributes, LPSECURITY_ATTRIBUTES lpThreadAttributes, BOOL bInheritHandles, DWORD dwCreationFlags, LPVOID lpEnvironment, LPCTSTR lpCurrentDirectory, LPSTARTUPINFO lpStartupInfo, LPPROCESS_INFORMATION lpProcessInformation );  BOOL CreateProcessWithLogonW( LPCWSTR lpUsername, LPCWSTR lpDomain, LPCWSTR lpPassword, DWORD dwLogonFlags, LPCWSTR lpApplicationName, LPWSTR lpCommandLine, DWORD dwCreationFlags, LPVOID lpEnvironment, LPCWSTR lpCurrentDirectory, LPSTARTUPINFO lpStartupInfo, LPPROCESS_INFORMATION lpProcessInfo ); </pre>
<b>Examples of Incorrect Code</b>	<pre> /* Create a security descriptor per "MSDN Creating a Security Descriptor for a New Object in C++" cited below, but specify permissions that grant anyone the ability to write. Then use this descriptor to create a new process. Example is large, but the main point is that an overly </pre>

```

    permissive security descriptor is
    being applied. */

    DWORD dwRes, dwDisposition;
    PSID pEveryoneSID = NULL,
    pAdminSID = NULL;
    PACL pACL = NULL;
    PSECURITY_DESCRIPTOR pSD = NULL;
    EXPLICIT_ACCESS ea[1];
    SID_IDENTIFIER_AUTHORITY
    SIDAuthWorld =
    SECURITY_WORLD_SID_AUTHORITY;
    SID_IDENTIFIER_AUTHORITY
    SIDAuthNT = SECURITY_NT_AUTHORITY;
    SECURITY_ATTRIBUTES sa;
    LONG lRes;
    HKEY hkSub = NULL;

    // Create a well-known SID for the
    // Everyone group.
    if(!
        AllocateAndInitializeSid(&SIDAuthWorld,
        1,
        SECURITY_WORLD_RID,
        0, 0, 0, 0, 0, 0,
        &pEveryoneSID)) { /* handle error
        */ }

    // Initialize an EXPLICIT_ACCESS
    // structure for an ACE.
    // The ACE will allow Everyone
    // full access
    ZeroMemory(&ea, 1 *
    sizeof(EXPLICIT_ACCESS));
    ea[0].grfAccessPermissions =
    KEY_ALL_ACCESS;
    ea[0].grfAccessMode = SET_ACCESS;
    ea[0].grfInheritance=
    NO_INHERITANCE;
    ea[0].Trustee.TrusteeForm =
    TRUSTEE_IS_SID;
    ea[0].Trustee.TrusteeType =
    TRUSTEE_IS_WELL_KNOWN_GROUP;
    ea[0].Trustee.ptstrName =
    (LPTSTR) pEveryoneSID;

    // Create a new ACL that contains
    // the new ACES.
    dwRes = SetEntriesInAcl(1, ea,
    NULL, &pACL);
    if (ERROR_SUCCESS != dwRes) { /* handle error */ }

    // Initialize a security
    // descriptor.

```

```

pSD = (PSECURITY_DESCRIPTOR)
LocalAlloc(LPTR,
SECURITY_DESCRIPTOR_MIN_LENGTH);
if (NULL == pSD) { /* handle
error */ }

if (!
InitializeSecurityDescriptor(pSD,
SECURITY_DESCRIPTOR_REVISION)) { /* /
* handle error */ }

// Add the ACL to the security
descriptor.
if (!
SetSecurityDescriptorDacl(pSD,
TRUE, // bDaclPresent flag
pACL,
FALSE)) // not a default DACL
{ /* handle error */ }

// Initialize a security
attributes structure.
sa.nLength = sizeof
(SECURITY_ATTRIBUTES);
sa.lpSecurityDescriptor = pSD;
sa.bInheritHandle = TRUE; // allows inheritance - would be
safer to set this FALSE

// Use the security attributes to
set the security descriptor
// when you create a key.
lRes =
RegCreateKeyEx(HKEY_CURRENT_USER,
"mykey", 0, "", 0,
KEY_READ | KEY_WRITE, &sa,
&hkSub, &dwDisposition);
printf("RegCreateKeyEx result %u
\n", lRes );

STARTUPINFO si;
PROCESS_INFORMATION pi;

ZeroMemory( &si, sizeof(si) );
si.cb = sizeof(si);
ZeroMemory( &pi, sizeof(pi) );
// Start the child process.
if( !CreateProcess( NULL, // No
module name (use command line).
TEXT("MyChildProcess"), // Command line.
&sa, // Use the security
descriptor we've created.
NULL, // Thread handle not
inheritable.

```

```

        FALSE, // Set handle inheritance
        to FALSE.
        0, // No creation flags.
        NULL, // Use parent's environment
        block.
        NULL, // Use parent's starting
        directory.
        &si, // Pointer to STARTUPINFO
        structure.
        &pi ) // Pointer to
        PROCESS_INFORMATION structure.
    ) { /* handle error */ }

```

### Examples of Corrected Code

```

/* Specifying a NULL security
descriptor to CreateProcess()
will apply a default security
descriptor to the new process.
Normally, this should be
reasonably restrictive and hence
reasonably secure. But check to
confirm specifics. */

STARTUPINFO si;
PROCESS_INFORMATION pi;

ZeroMemory( &si, sizeof(si) );
si.cb = sizeof(si);
ZeroMemory( &pi, sizeof(pi) );
// Start the child process.
if( !CreateProcess( NULL, // No
module name (use command line).
TEXT("MyChildProcess"), // Command line.
NULL, // Process handle not
inheritable - default security
descriptor used
NULL, // Thread handle not
inheritable.
FALSE, // Set handle inheritance
to FALSE.
0, // No creation flags.
NULL, // Use parent's environment
block.
NULL, // Use parent's starting
directory.
&si, // Pointer to STARTUPINFO
structure.
&pi ) // Pointer to
PROCESS_INFORMATION structure.
) { /* handle error */ }

```

### Source Reference

- [http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dllproc/base/processes\\_and\\_threads.asp](http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dllproc/base/processes_and_threads.asp)<sup>2</sup>

<b>Recommended Resources</b>	<ul style="list-style-type: none"> <li>• <a href="#">MSDN Processes and Threads<sup>3</sup></a></li> <li>• <a href="#">MSDN CreateProcess reference<sup>4</sup></a></li> <li>• <a href="#">MSDN Creating a Security Descriptor for a New Object in C++<sup>5</sup></a></li> <li>• <a href="#">MSDN EXPLICIT_ACCESS structure<sup>6</sup></a></li> </ul>				
<b>Discriminant Set</b>	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="padding: 5px;"><b>Operating System</b></td><td> <ul style="list-style-type: none"> <li>• Windows</li> </ul> </td></tr> <tr> <td style="padding: 5px;"><b>Languages</b></td><td> <ul style="list-style-type: none"> <li>• C</li> <li>• C++</li> </ul> </td></tr> </table>	<b>Operating System</b>	<ul style="list-style-type: none"> <li>• Windows</li> </ul>	<b>Languages</b>	<ul style="list-style-type: none"> <li>• C</li> <li>• C++</li> </ul>
<b>Operating System</b>	<ul style="list-style-type: none"> <li>• Windows</li> </ul>				
<b>Languages</b>	<ul style="list-style-type: none"> <li>• C</li> <li>• C++</li> </ul>				

## Cigital, Inc. Copyright

Copyright © Digital, Inc. 2005-2007. Digital retains copyrights to this material.

Permission to reproduce this document and to prepare derivative works from this document for internal use is granted, provided the copyright and “No Warranty” statements are included with all reproductions and derivative works.

For information regarding external or commercial use of copyrighted materials owned by Digital, including information about “Fair Use,” contact Digital at [copyright@digital.com<sup>1</sup>](mailto:copyright@digital.com).

The Build Security In (BSI) portal is sponsored by the U.S. Department of Homeland Security (DHS), National Cyber Security Division. The Software Engineering Institute (SEI) develops and operates BSI. DHS funding supports the publishing of all site content.

---

1. <mailto:copyright@digital.com>